Working Draft   T10
American National   Project 1380-D
Standard

Information Technology -
SCSI on Scheduled Transfer Protocol (SST)

T10 Technical Editor: Donald D. Woelz
                           GENROCO, Inc.
                           255 Info Hwy
                           Slinger, WI 53086
                           USA

                           Telephone: 262-644-2505
                           Facsimile: 262-644-6667
                           Email: don@genroco.com

POINTS OF CONTACT:

T10 Chair
John B. Lohmeyer
LSI
4420 ArrowsWest Drive
Colo Spgs, CO 80907-3444
Tel: (719) 533-7560
Fax: (719) 593-7183
Email: lohmeyer@t10.org

T10 Vice-Chair
George O. Penokie
IBM
3605 Highway 52 N. MS Z9V
Rochester, MN 55901
Tel: (507) 253-5208
Fax: (507) 253-2880
Email: gop@us.ibm.com

NCITS Secretariat

NCITS Secretariat
1250 Eye Street, NW Suite 200
Washington, DC 20005

Telephone: 202-737-8888
Facsimile: 202-638-4922
Email: ncits@itic.org

T10 Reflector   Internet address for subscription to the T10 reflector: majordomo@t10.org
The message body should be: subscribe t10
Internet address for distribution via T10 reflector: T10@t10.org
Internet address to unsubscribe from the T10 reflector: majordomo@t10.org
The message body should be: unsubscribe t10

Document Distribution

Global Engineering
15 Inverness Way East
Englewood, CO 80112-5704

Telephone: 303-792-2181 or
Toll Free: 800-854-7179
Facsimile: 303-792-2192

ABSTRACT

This standard specifies a mapping for SCSI commands, data transfers, and responses using the Scheduled Transfer (ST) protocol (*ANSI NCITS.337-2000*, Information Technology - Scheduled Transfer (ST)) as the lower level protocol. The ST protocol is defined for a variety of network media including Ethernet, ATM, and HIPPI-6400. The mapping of SCSI storage data onto the ST protocol layer enables storage area networking (SAN) implementations on a wide variety of common network infrastructures.

Patent Statement

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Comments on Rev 08b

This is a preliminary standard undergoing lots of changes. Many of the additions are just place holders, or are put there to stimulate discussion. Hence, do not assume that the items herein are correct, or final – everything is subject to change. This page tries to outline where we are; what has been discussed and semi-approved, and what has been added or changed recently and deserves your special attention. This summary relates to changes since the previous revision. Also, previous open issues are outlined with a single box, new open issues ones are marked with a double bar on the left edge of the box.

Changes are marked with margin bars so that changed paragraphs are easily found, and then highlights mark the specific changes. The list below just describes the major changes, for detail changes please compare this revision to the previous revision. **The major technical changes are printed in bold.**

Please help us in this development process by sending comments, corrections, and suggestions to the Technical Editor, Don Woelz, of GENROCO, Inc., at don@genroco.com. If you would like to address the whole group working on this standard, then send the comment(s) to hippi@nsco.network.com.

1. This revision of this standard is the same as revision 8a except that all of the editing change highlighting has been removed.

American National Standard
for Information Technology

SCSI on Scheduled Transfer Protocol (SST)

Secretariat
Information Technology Industry Council (ITI)

Approved   , 2001

American National Standards Institute, Inc.

Abstract

This standard specifies a mapping for SCSI commands, data transfers, and responses using the Scheduled Transfer (ST) protocol (*ANSI NCITS.337-2000*, Information Technology - Scheduled Transfer (ST)) as the lower level protocol. The ST protocol is defined for a variety of network media including Ethernet, ATM, and HIPPI-6400. The mapping of SCSI storage data onto the ST protocol layer enables storage area networking (SAN) implementations on a wide variety of common network infrastructures.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, no such claims has been been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

CONTENTS

Tables

Figures

Foreword (This foreword is not part of this standard)

This standard specifies a mapping for SCSI commands, data transfers, and responses using the Scheduled Transfer (ST) protocol (*ANSI NCITS.337-2000*, Information Technology - Scheduled Transfer (ST)) as the lower level protocol. The ST protocol is defined for a variety of network media including Ethernet, ATM, and HIPPI-6400. The mapping of SCSI storage data onto the ST protocol layer enables storage area networking (SAN) implementations on a wide variety of common network infrastructures.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the National Committee for Information Technology Standards, 1250 Eye Street, NW, Suite 200, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by NCITS. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, the NCITS had the following members:

James D. Converse, Chair
Donald C. Loughry, Vice-chair
Joanne M. Flanagan, Secretary

Organization Represented	*Name of Representative*
American Nuclear Society	Geraldine C. Main
AMP, Inc	Edward Kelly
Apple Computer	Karen Higginbottom
Association of the Institute for Certification of Professionals	Kenneth Zemrowski
AT&T/NCR	Thomas W. Kern
Boeing Company	Catherine Howells
Bull HN Information Systems, Inc.	William George
Compaq Computer Corporation	James Barnes
Digital Equipment Corporation	Delbert Shoemaker
Eastman Kodak	James D. Converse
GUIDE International	Frank Kirshenbaum
Hewlett-Packard	Donald C. Loughry
Hitachi America, Ltd	John Neumann
Hughes Aircraft Company	Harold L. Zebrack
IBM Corporation	Joel Urman
National Communication Systems	Dennis Bodson
National Institute of Standards and Technology	Robert E. Roundtree
Northern Telecom, Inc.	Mel Woinsky
Neville & Associates	Carlton Neville
Recognition Technology Users Association	Herbert P. Schantz
Share, Inc.	Gary Ainsworth
Sony Corporation	Michael Deese
Storage Technology Corporation	Joseph S. Zajaczkowski
Sun Microsystems	Scott Jameson
3M Company	Eddie T. Morioka
Unisys Corporation	John L. Hill
US Department of Defense	William C. Rinehuls
US Department of Energy	Alton Cox
US General Services Administration	Douglas Arai
Wintergreen Information Services	Joun Wheeler
Xerox Corporation	Dwight McBain

Technical Committee T10 on SCSI Interfaces, which reviewed this standard, had the following participants:

John B. Lohmeyer, Chair

Lawrence J. Lamers, Vice-chair
Ralph O. Weber, Secretary

I. D. Allan	T. J. Kulesza	D. Piper
P. D. Aloisi	A. Littlewood	G. Porter
R. Bellino	B. Masterson	R. Reisch
C. Brill	W. P. McFerrin	J. R. Sims
R. Cummings	J. McGrath	R. N. Snively
Z. Daggett	P. McLean	G. R. Stephens
J. Dambach	G. McSorley	C. Tashbook
R. Freeze	P. Mercer	P. Tobias
E. A. Gardner	G. Milligan	T. Totani
D. Guss	C. Monia	D. Wagner
K. J. Hallam	D. Moore	R. Wagner
E. Haske	I. Morrell	D. Wallace
P. Jadeja	N. Nadershahi	J. L. Williams
P. Johansson	C. Nieves	M. Wingard
G. Johnsen	E. Oetting	K. Wolfgang
S. Jones	D. Pak	A.Yang
C. Kephart	K. W. Parker	
M. Kuhlmeyer	G. Penokie	

Task Group T11.1 on the High-Performance Parallel Interface, which developed this standard, had the following participants:

Roger Ronald, Chairman
Don Woelz, Vice Chairman and SST Technical Editor

B. Allen	M. Howard	T. Plunkett
S. Bailey	L. Huff	S. Rieb
J. Bhat	D. Hyer	E. Salo
M. Boorman	R. Hyerle	D. Sanders
G. Boyd	D. Johnson	C. Satterlee
E. Brady	A. Kelley	A. Singla
B. Breuer	V. Kengeri	W. Smith
E. Cady	C. Lindahl	W. St. John
G. Chesson	B. Lynn	L. Stewart
J. Chung	A. Maccabe	S. Suen
R. Cummings	M. McGowen	T. Torrico
C. Davidson	R. Newhall	T. Truong
M. Doppke	J. Nordman	T. Ut sumi
N. Droux	C. Pan	A. van Praag
J. Evans	R. Pearson	R. Willard
T. Gilbert	I. Philp	J. Young
D. Hagerman	C. Pick	
P. Harper	J. Pinkerton	
J. Hoffman	J. Pittet	

Introduction

The SCSI family of standards is developed by NCITS T10 to facilitate the use of the SCSI command sets for different types of devices over a variety of physical interconnects. The architectural document of the family of standards is ANSI X3.270-1996 Information Technology - SCSI Architecture Model (SAM).

The SCSI on ST (SST) standard defines a transport protocol within the SCSI family of standards as shown in figure 1. The physical interconnects to which the SST protocol may attach are not defined within this standard, but rather, are any interconnects or other protocols on which the basic ST protocol may operate (see ANSI NCITS.337-2000).

Clause 1 outlines the scope of this standard.

Clause 2 provides a cross reference to other relevant standards.

Clause 3 provides definitions and conventions used in this standard.

Clause 4 presents an overview of this standard.

Clause 5 describes how SCSI operations are structurally mapped onto ST operations in the SST protocol.

Clause 6 defines specific message formats for the SST protocol.

Clause 7 defines error recovery procedures for the SST protocol.

Common Access Method

Architectural Model

SCSI Device-Type Specific Command Sets

Shared Command Set (for all SCSI device types)

Transport Protocols (e.g. SST)

Physical Interconnects
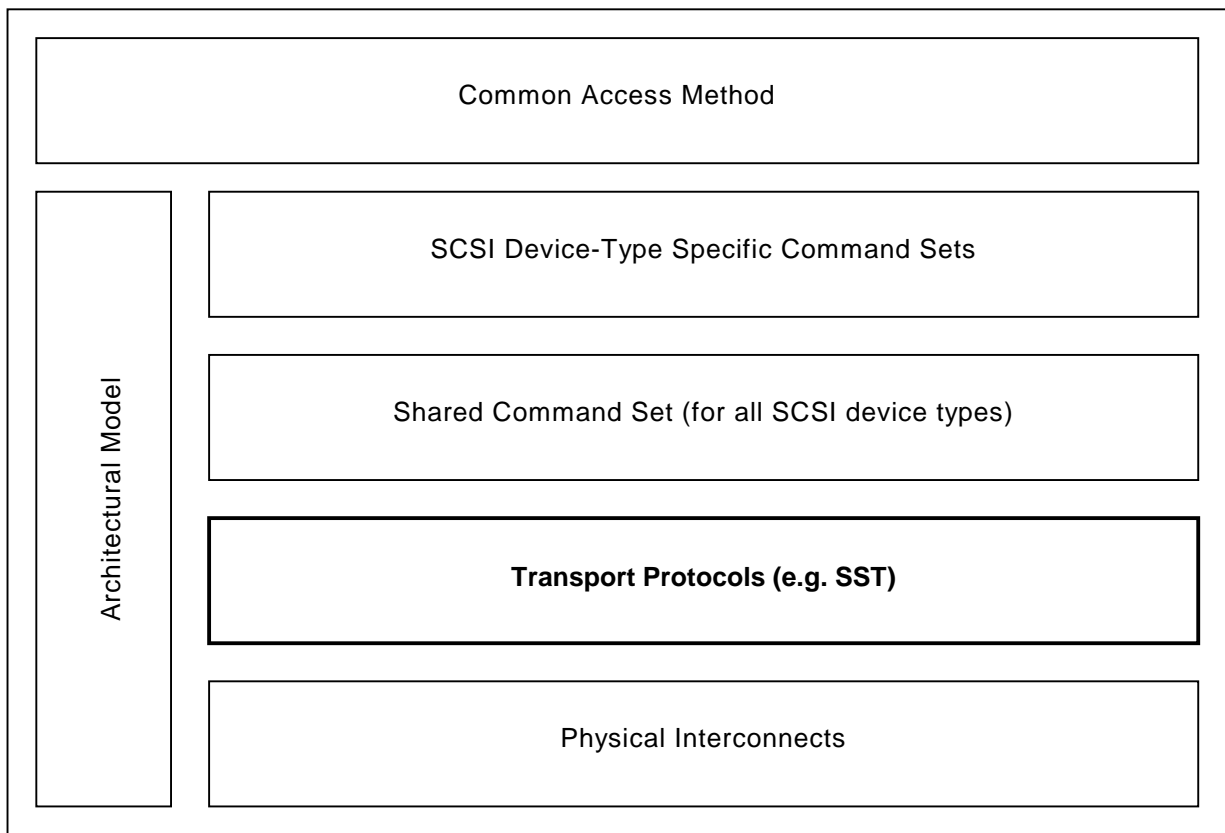
Figure 1 – Relationship of SST to SAM

**American National Standard
for Information Technology –**

SCSI on Scheduled Transfer Protocol (SST)

1   Scope

This standard specifies a mapping for SCSI commands, data transfers, and responses using the Scheduled Transfer (ST) protocol (*ANSI NCITS.337-2000*, Information Technology - Scheduled Transfer (ST)) as the lower level protocol. The ST protocol is defined for a variety of media including ATM (Asynchronous Transfer Mode), Ethernet (IEEE 802), and HIPPI (both 800/1600 and 6400 megabit/second High-Performance Parallel Interface networks). The mapping of SCSI storage data onto the ST protocol layer enables storage area networking (SAN) implementations on a wide variety of common network infrastructures.

Specifications are included in this standard for

— connection management,

— device management,

— task management,

— SCSI command service requests using ST Request_To_Send, Request_To_Receive, and Nop operations,

— SCSI data delivery requests using the ST Clear_To_Send operation,

— SCSI data delivery actions using ST Data operations,

— SCSI command service responses using the ST Put operation to a Persistent Memory Region,

— SCSI I/O operations using ST Read, Write, Nop, and Put sequences, and

— aborting connections and operations.

2   Normative references

The following standards contain provisions that, through reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the following documents can be obtained from ANSI: Approved ANSI standards, approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT) and approved foreign standards (including BSI, JIS, and DIN). For further information, contact ANSI Customer Service Department at 212-642-4900 (phone) 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.   Additional availability contact information is provided below as needed.

2.1   Approved references

ANSI X3.131-1994, Information Systems – Small Computer System Interface – 2 (SCSI-2)

ANSI X3.270-1996, Information Systems – SCSI-3 Architecture Model (SAM)

ANSI NCITS.337-2000, Information Technology - Scheduled Transfer (ST)

2.2   References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the standard or regarding availability, contact the relevant standards body or other organization as indicated. For information about obtaining copies of this standard or for more information on the current status of the standard, contact National Committee for Information Technology Standards, 1250 Eye Street, NW, Suite 200, Washington, DC 20005, 202-626-5746.

NCITS T10, Project 1157-D, SCSI Architecture Model - 2 (SAM-2)

NCITS T10, Project 1236-D, SCSI Primary Commands - 2 (SPC-2)

3   Definitions and conventions

3.1   Definitions

For the purposes of this standard, the following definitions apply.

**3.1.1
Block**
An ordered set of one or more STUs (see 3.1.18) within an ST Read, Write, or Put sequence.

**3.1.2
Buffer Index (Bufx)**
A 32-bit parameter identifying the starting address of a data buffer.

**3.1.3
Data Channel**
The logical channel that carries the data payload.

**3.1.4
Data operation**
A data transmission consisting of a Schedule Header and up to 4 gigabytes of user payload.

**3.1.5
Destination**
The end device that receives an operation or data.

**3.1.6
device server**
An object within the logical unit which executes SCSI tasks and enforces the rules for task management (see ANSI X3.270).

**3.1.7
expose**
Enable a memory region for data operations.

3.1.8
initiator
An end device containing SCSI application clients which originate SCSI device service and SCSI task management requests to be processed by a SCSI device. Note: both SAM and ST define the term initiator. This standard uses the term initiator according to the SAM definition, unless otherwise indicated.

3.1.9
Key
A local identifier used to select and validate operations.

3.1.10
logical unit
A target resident entity that implements a device model and executes SCSI commands sent by an application client (see ANSI X3.270).

3.1.11
lower-layer protocol (LLP)
A protocol below the ST Protocol, e.g., a physical layer.

3.1.12
Offset
A parameter specifying the data's starting point relative to the start of a Bufx (see 3.1.2).

3.1.13
Opaque data
Four bytes of Source ULP to Destination ULP peer-to-peer information carried in a Data operation's Schedule Header separately from the data payload.

3.1.14
operation
The procedure defined by the parameters in a Schedule Header, and any payload associated with that Schedule Header. The code in the Schedule Header's "Op" field identifies the operation's name/function.

3.1.15
persistent memory
Memory that is maintained for multiple ST Put operations.

3.1.16
Put
An operation to write data into a persistent memory region on a remote end device.

3.1.17
Scheduled Transfer
An information transfer, normally used for bulk data movement, where the end devices prearrange the transfer using the protocol defined in the Scheduled Transfer standard (see ANSI NCITS.337-2000).

3.1.18
Scheduled Transfer Unit (STU)
The data payload portion of a Data operation. STUs are the basic components of Blocks (see 3.1.1) and are the smallest units transferred.

3.1.19
SCSI device
A device that originates or services SCSI commands (see ANSI X3.270).

3.1.20
sequence
An ordered group of ST operations providing a particular function, e.g., Read, Write, Put, etc., between an ST Initiator and an ST Responder. The roles of ST Initiator and ST Responder are constant for all operations in the sequence.

3.1.21
SST endpoint
A network addressable port which supports the SST protocol.

3.1.22
SST I/O operation
An unlinked SCSI command, a series of linked SCSI commands, or a task management function. (see ANSI X3.270).

3.1.23
SST Read operation
ST sequences which execute a SCSI command with data transfer from a SCSI target to SCSI initiator.

3.1.24
SST Virtual Connection (SSTVC)
An extension to the ST Virtual Connection that is opened on the SST reserved ULP port number with additional parameters and managed according to the rules defined in 5.2.

3.1.25
SST Write operation
ST sequences which execute a SCSI command with data transfer from a SCSI initiator to SCSI target.

3.1.26
SST zero length operation
ST sequences which execute a SCSI command with no data transfer between the SCSI initiator and SCSI target.

3.1.27
SSTVC Initiator
An end device that starts the sequence of operations to create an SSTVC. Note: The term SSTVC Initiator should not be confused with the term initiator (3.1.7). An SSTVC Initiator refers to the initiator role in forming an ST Virtual Connection, where initiator refers to a SCSI initiator, defined by SAM.

3.1.28
SSTVC Responder
An end device that responds to the SSTVC Initiator to create an SSTVC.

3.1.29
ST Buffer Size
The unit of memory addressed by ST for Bufx and Offset calculations and expressed as $2^{Bufsize}$ bytes.

3.1.30
Status Persistent Memory Region (SPMR)
An ST Persistent Memory Region established by the SSTVC Initiator to contain SCSI status information (see 5.2.1)

3.1.31
tag
The initiator-specified component of the task identifier (see ANSI X3.270).

3.1.32
target
An end device which receives SCSI commands and directs such commands to one or more logical units for execution.

3.1.33
task
An object within the logical unit representing the work associated with a SCSI command or a group of linked SCSI commands (see 4.1).

3.1.34
task identifier
The information uniquely identifying a task (see ANSI X3.270).

3.1.35
Transfer
An ordered set of one or more Blocks (see 3.1.1) within a Scheduled Transfer (see 3.1.17).

3.1.36
upper-layer protocol (ULP)
The protocol above ST. A ULP could be implemented in hardware or software, or could be distributed between the two.

3.1.37
Virtual Connection
A bi-directional logical connection used for Scheduled Transfers between two end devices. A Virtual Connection contains a logical Control Channel and one or more logical Data Channels in each direction.

3.2   Editorial conventions

Conditions, sequence parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., Block, Transfer). Any lowercase uses of these words have the normal technical English meaning.

Some field names used in this standard are imported from the SCSI family of standards, and others are imported from the ST standard. The naming conventions of items imported from these standards are consistent with the conventions used in the standards from which they are drawn.

Multiword parameters and field names are joined with an underscore, e.g., Clear_To_Send. A parameter associated with a particular end device uses a single letter prefix and a hyphen as a joiner, e.g., I-Id denoting the initiator's Id.

Operations contained within <…> are conditional, and may not occur.

If a conflict arises between text, tables, or figures, the order of precedence to resolve the conflict is text; then tables; and finally figures. Not all tables or figures are fully described in the text. Tables show data format and values.

All numbers are represented as unsigned integers.

Numbers that are not immediately followed by lower-case b or h are decimal values.

Numbers immediately followed by lower-case b (e.g. 0101b) are binary values. In all of the figures, tables, and text of this document, the most significant bit of a binary quantity is shown on the left side.

Numbers or upper case letters immediately followed by lower-case h (e.g. FA23h) are hexadecimal values.

3.3   Acronyms and other abbreviations

ACA   Auto Contingent Allegiance
ATM   Asynchronous Transfer Mode
CDB   Command Descriptor Block
CTS   Clear to Send
HIPPI   High-Performance Parallel Interface
id   identifier
IEEE   Institute of Electrical and Electronic Engineers
LEN   Length
LLP   lower-layer protocol
LUN   Logical Unit Number
MB   megabyte (i.e., 10^6 bytes)
num   number, as in B_num
PMR   Persistent Memory Region
RSP   Response
RTR   Request to Receive
RTS   Request to Send
SAM   SCSI Architecture Model
SCSI   Small Computer System Interconnect
SNS   Sense
SPMR   Status Persistent Memory Region
SST   SCSI on Scheduled Transfer Protocol
SSTVC   SST Virtual Connection
ST   Scheduled Transfer
STU   Scheduled Transfer Unit (pronounced as "stew")
TM   task management
ULP   upper-layer protocol

3.4 Keywords

**3.4.1
invalid**
A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as error.

**3.4.2
ignored**
A keyword used to describe a bit, byte, word, field or code value that shall not be examined. The bit, byte, word, field or code value has no meaning in the specified context.

**3.4.3
mandatory**
A keyword indicating an item that is required to be implemented as defined in this standard.

**3.4.4
may**
A keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

**3.4.5
may not**
A keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

**3.4.6
optional**
A keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined by this standard is implemented, then it shall be implemented as defined in this standard.

**3.4.7
reserved**
A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

**3.4.8
shall**
A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard. This standard prescribes no specific response by a component if it receives information that violates a mandatory behavior.

**3.4.9
should**
A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase "it is strongly recommended".

4   Overview

4.1   Structure and concepts

ST is a data transfer protocol which may be implemented on a wide variety of Lower Layer Protocols (LLP), which supports a connection-oriented data transfer model with multiple outstanding data transfers per connection. SST layers only on ST.

Three kinds of functional management are defined by the SST specification:

— Connection management;

— Device management;

— Task management.

SST uses one or more distinct ST sequences (e.g. Read sequence, Write sequence, PMR Put sequence), which are logically associated to perform a complete SST I/O operation.

The correspondence between elements of the SCSI protocol, elements of the SST protocol, and elements of the ST protocol on which SST is built is shown in table 1.

Table 1 – Functional correspondence between SCSI, SST, and ST operations

SCSI	SST	ST
Device service request	Request_To_Send or Request_To_Receive or Request_Zero_Length	Request_To_Send operation Request_To_Receive operation Nop operation
Data delivery request	Clear_To_Send operation	Clear_To_Send operation
Data delivery action	Data operation (STU)	Data operation
Device service response	Status Put	Put operation
Unlinked command execution	SST I/O operation	Read sequence and Put sequence or Write sequence and Put sequence or Nop operation and Put sequence
Linked command execution	SST I/O operation	A sequence of: Read sequence and Put sequence or Write sequence and Put sequence or Nop operation and Put sequence
Task management function	SST I/O operation or End sequence	Nop operation and Put operation or End sequence

5   SST characteristics

5.1   Common ST operation characteristics

All ST Control operations used in the SST protocol shall have the ST Interrupt flag bit set.

Non-final ST Data operations of Read sequences and Write sequences shall have the ST Interrupt Flag bit set to 0b, Last Flag bit set to 0b, and Silent Flag bit set to 1b.

The final ST Data operation of Read sequences and Write sequences shall have the ST Interrupt Flag bit set to 1b, Last Flag bit set to 1b, and Silent Flag bit set to 0b.

All ST Data operations for an SST connection shall be sent on the ST Data Channel specified in the SST connection setup protocol (see 5.2). ST Data Channel flag bits shall be appropriately set to reflect this.

The SST protocol does not use the operation pairs for reliable data movement (see ST). Instead, entire SST I/O operations are retried as described in 7.4, SST I/O operation timeouts.

All multibyte quantities are formed using big-endian ordering.

5.2   Connection management

An SST Virtual Connection (SSTVC) is an extension of the basic ST Virtual Connection. To create an SSTVC, a pair of end devices create an ST Virtual Connection, then extend that into an SSTVC.

Note that an SSTVC may be forward, reverse, or symmetrical. If the SSTVC is a forward SSTVC, then only the SSTVC Initiator is a SCSI initiator. If the SSTVC is a reverse SSTVC, then only the SSTVC Responder is a SCSI initiator. If the SSTVC is a symmetrical SSTVC, then both the SSTVC Initiator and the SSTVC Responder are SCSI initiators. The SSTVC_Parameters operation contains information indicating the ability of the SSTVC Initiator or Responder to be a SCSI initiator (See 6.1).

The usual SSTVC is forward, which corresponds to a SCSI initiator (e.g. a host) connecting to a SCSI target (e.g. a storage device). It is not anticipated that reverse SSTVCs will ever occur, since this corresponds to a SCSI target connecting to a SCSI initiator, but a reverse SSTVC is not prohibited. A symmetrical SSTVC may occur when two processor devices are using SCSI to communicate with each other, as in some clustering applications, or possibly among storage devices performing third party SCSI operations, or other functions which result in SCSI devices performing both initiator and target roles.

Figure 2 shows a typical SSTVC connection setup sequence for a symmetrical SSTVC. The setup of an SSTVC is described in more detail in 5.2.1.

SSTVC Initiator   **SSTVC Responder**

Initiate connection setup   ST Request_Connection
                                                        ───────────────────────────────────▶
                                                             ST Connection_Answer
                                                        ◀───────────────────────────────────

Exchange SSTVC   SSTVC Initiator SSTVC_Parameters
Parameters   ───────────────────────────────────▶
                                              SSTVC Responder SSTVC_Parameters
                                                        ◀───────────────────────────────────

Responder requests SPMR   ST Request_Memory_Region
(for a symmetrical or   ◀- - - - - - - - - - - - - - - - - - -
forward SSTVC)   ST Memory_Region_Available
                                                        - - - - - - - - - - - - - - - - - - -▶

SSTVC Initiator requests   ST Request_Memory_Region
SPMR (for a symmetrical   - - - - - - - - - - - - - - - - - - -▶
or reverse SSTVC)   ST Memory_Region_Available
                                                        ◀- - - - - - - - - - - - - - - - - - -

Figure 2 – SSTVC Connection setup sequence

5.2.1   Connection setup

An SSTVC shall be requested by issuing an ST Request_Connect operation to the well known port for SST with the Out_Of_Order (O) flag bit set to 0b. SST requires that out-of-order operation not be requested in order to support the ST Clear_To_Send operation accounting protocol described in 5.4.1. The well known port for SST shall be as defined by the Internet Assigned Numbers Authority (IANA) and as published at http://www.iana.org/.

If the SSTVC Responder is willing to accept the SSTVC, it shall respond with an ST Connection_Answer operation with the Out_Of_Order (O) flag bit set to 0b. Otherwise the SSTVC

Responder shall reject the SSTVC with an ST Connection_Answer operation with the Reject (R) flag bit set to 1b.

If the SSTVC is accepted by the SSTVC Responder, the SSTVC Initiator shall issue an SSTVC_Parameters operation with its SSTVC parameters in the option payload (see 6.1).

If the SSTVC parameters are unacceptable to the SSTVC Responder, then the SSTVC Responder shall disconnect the SSTVC using the standard ST Virtual Connection teardown sequence. If the SSTVC parameters are acceptable to the SSTVC Responder, then the SSTVC Responder shall issue an SSTVC_Parameters operation with its SSTVC parameters in the option payload (see 6.1). If the SSTVC parameters specified by the SSTVC Responder are unacceptable to the SSTVC Initiator, then the SSTVC Initiator shall disconnect the SSTVC using the standard ST Virtual Connection teardown sequence.

If the parameters in the SSTVC Initiator SSTVC_Parameters operation indicate that the SSTVC Initiator can function as a SCSI initiator and the SSTVC Responder intends to function as a SCSI target, then the SSTVC Responder shall use an ST Request_Memory_Region operation to request that the SSTVC Initiator expose a Persistent Memory Region, called the Status Persistent Memory Region (SPMR), for the ST Data Channel specified in the SSTVC_Parameters option payload. The SPMR size shall be specified in the SSTVC_Parameters operation issued by the SSTVC Initiator. The SSTVC Initiator shall expose the SPMR and respond with an appropriate Memory_Region_Available operation. The initial Offset of the SPMR shall be a multiple of 512 bytes, the SPMR segment size.

When the SSTVC_Parameters operation issued by the SSTVC Responder is received by the SSTVC Initiator and the connection parameters specify that the SSTVC Responder can function as a SCSI initiator and the SSTVC Initiator intends to function as a SCSI target, then the SSTVC Initiator shall request that the SSTVC Responder expose an SPMR as described above.

An SSTVC endpoint shall not initiate any SCSI operations until the entire SSTVC Connection setup sequence has been successfully completed.

5.2.2   Connection setup timeouts.

Operation pairs shall be guarded by timeouts in the four cases listed below. If any timeout occurs, the SSTVC shall be disconnected using the ST virtual connection teardown sequence. Connections torn down in this manner may be retried. The exact duration of the timer is implementation specific. However, the SST task timeout value (see 7.2) is a recommended timer duration. Connection error recovery shall be done according to clause 7.

— The SSTVC Responder shall maintain a timer between issuance of the ST Connection_Answer and receipt of the SSTVC Initiator SSTVC_Parameters.
— The SSTVC Initiator shall maintain a timer between issuance of the SSTVC Initiator SSTVC_Parameters and receipt of the Responder SSTVC_Parameters.
— The SSTVC Initiator shall maintain a timer between issuance of the SSTVC Initiator SSTVC_Parameters and receipt of the ST Request_Memory_Region if this is a forward or symmetrical SSTVC.
— The SSTVC Responder shall maintain a timer between issuance of the SSTVC Responder SSTVC_Paramaters and receipt of the ST Request_Memory_Region if this is a reverse or symmetrical SSTVC.

Note that other operation pairs used in SSTVC connection setup from ST Request_Connection to ST Connection_Answer and from ST Request_Memory_Region to ST Memory_Region_Available are guarded by ST timeouts.

5.2.3   Connection disconnect

An SSTVC shall be disconnected using the ST Virtual Connection teardown sequence (see ST).

5.3   Device management

Each SCSI command within an SST I/O operation is performed as:

— an ST Read, Write or Nop sequence followed by;

— an ST Put sequence to the SPMR containing status.

Note that SCSI commands may be linked and are executed serially.

An SST I/O operation is begun for each unlinked SCSI command or the first of each set of linked SCSI commands presented as SCSI Execute service requests. The SST protocol performs an SST Write operation, SST Read operation, or SST zero length operation for the execution of each SCSI command within an SST I/O operation.

If the command is linked to another command, then the ST Put payload shall contain the proper SCSI status indicating that another command will be executed. The initiator shall continue the same SST I/O operation with an ST Request_To_Send, ST Request_To_Receive, or SST Request_Zero_Length operation beginning the next SCSI command. All SCSI commands linked in the SST I/O operation except the last shall be executed in the manner described above.

Note that when an SST I/O operation executes more than one linked SCSI command, the same tag shall be used for ST sequences in the SST I/O operation.

The number of SST I/O operations that may be active at one time depends on the queuing capabilities of the particular SCSI devices and the number of concurrent transactions supported by the SST endpoints.

5.3.1   SST Write operation

The initiator starts an SST Write operation by issuing an ST Request_To_Send operation (see 6.1).

The ST Request_To_Send operation shall have an option payload including command control flags, addressing information, and the SCSI Command Descriptor Block (CDB) as described in 0.

The ST Request_To_Send operation is the Execute Command service request and starts the SST Write operation. The SST I/O operation in which the SST Write operation is executed is identified by the initiator-specified tag qualified by the ST Virtual Connection. The tag shall be used as the ST I-id field for the ST Write sequence.
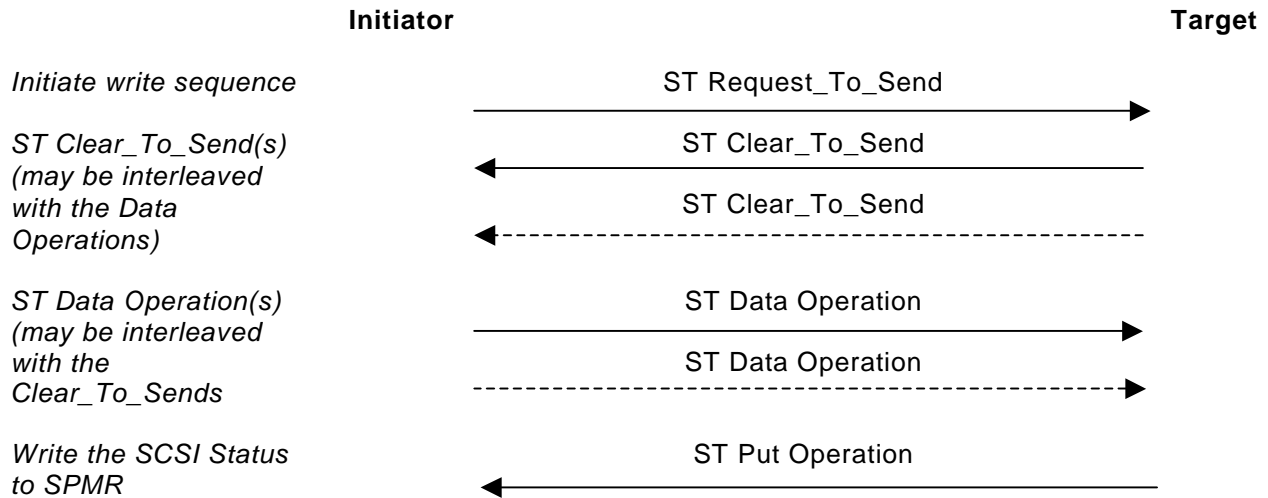
Initiator   **Target**

Initiate write sequence   ST Request_To_Send
                                        ───▶

ST Clear_To_Send(s)   ST Clear_To_Send
(may be interleaved   ◀───
with the Data   ST Clear_To_Send
Operations)   ◀- -

ST Data Operation(s)   ST Data Operation
(may be interleaved   ───▶
with the   ST Data Operation
Clear_To_Sends   -▶

Write the SCSI Status   ST Put Operation
to SPMR   ◀───

Figure 3 – SST Write operation

When the device server for the command transferred by the ST Request_To_Send operation has determined that a data transfer is required and is prepared to request the data delivery service, the data shall be transferred as described below. The amount of data transferred shall be the minimum of:

— the transfer length specified in the ST Request_To_Send operation which initiated the task; or,

— the amount of data the device server requires to be transferred as specified in the SCSI CDB.

The target shall issue one or more ST Clear_To_Send operations for the amount of data that it is prepared to receive.

The initiator shall then respond with one or more ST Data operations to satisfy the outstanding ST Clear_To_Send operations received from the target.

When the device server is prepared to receive additional data, the target may issue one or more additional ST Clear_To_Send operations to which the initiator responds with additional ST Data operations until the required amount of data has been transferred.

After all of the data has been transferred, the device server shall return the Execute Command service response to the initiator by requesting that the target issue an ST Put sequence to the SPMR established in the SSTVC setup protocol (see 5.2.1). The address (Bufx and Offset) for this ST Put shall be computed from the initiator-specified tag used as the ST I-id for the ST Write sequence (see 6.3). This ST Put sequence shall contain the SCSI status. If an exception condition has been detected, then it shall also contain the SCSI REQUEST SENSE information and the SST Response information that describes the condition. The SST Status Put shall terminate the command. The SCSI logical unit determines whether additional linked commands will be performed in the SST I/O operation. If this is the last or only command executed in the SST I/O operation, then the SST I/O operation and the task shall be terminated.

5.3.2   SST Read operation

The initiator starts an SST Read operation by issuing an ST Request_To_Receive operation (see 6.1).

The ST Request_To_Receive operation shall have an option payload including command control flags, addressing information, and the SCSI CDB as described in 0.

The ST Request_To_Receive operation is the Execute Command service request and starts the SST Read operation. The SST I/O operation in which the SST Read operation is executed is identified by the initiator-specified tag qualified by the ST Virtual Connection. The tag shall be used as the ST I-id field for the ST Read sequence.

Initiator   **Target**

Initiate read sequence by   ST Request_To_Receive
asking responder to write   ————————————————————————▶

Responder Initiates write   ST Request_To_Send
sequence   ◀————————————————————————

ST Clear_To_Send(s)   ST Clear_To_Send
(may be interleaved with   ————————————————————————▶
the Data Operations)   ST Clear_To_Send
    --▶

ST Data Operation(s) (may   ST Data Operation
be interleaved with the   ◀————————————————————————
Clear_To_Send(s))   ST Data Operation
    ◀--

Write the SCSI Status to   ST Put Operation
SPMR   ◀————————————————————————

Figure 4 – SST Read operation

When the device server for the command transferred by the ST Request_To_Receive operation has determined that a data transfer is required and is prepared to request the data delivery service, the data shall be transferred as described below. The amount of data transferred shall be the minimum of:

— the transfer length specified in the ST Request_To_Receive operation which initiated the task; or,

— the amount of data the device server requires to be transferred as specified in the SCSI CDB.

The target shall respond to the ST Request_To_Receive operation with an ST Request_To_Send operation. Data is then transferred from the target to the initiator in response to ST Clear_To_Send operations sent from the initiator to the target as appropriate for an ST Read sequence.
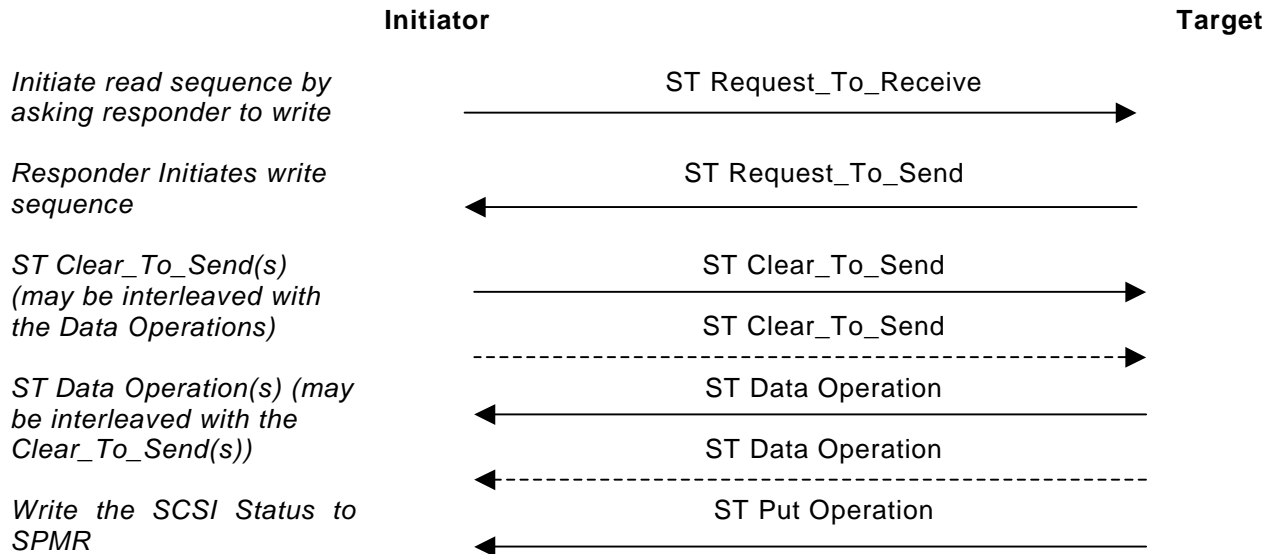
After all of the data has been transferred, the device server shall return the Execute Command service response to the initiator by requesting that the target issue an ST Put sequence to the SPMR established in the SSTVC setup protocol (see 5.2.1). The address (Bufx and Offset) for this ST Put shall be computed from the initiator-specified tag used as the ST I-id for the ST Read sequence (see 6.3). This ST Put sequence shall contain the SCSI status. If an exception condition has been detected, then it shall also contain the SCSI REQUEST SENSE information and the SST Response information that describes the condition. The SST Status Put shall terminate the command. The SCSI logical unit determines whether additional linked commands will be performed in the SST I/O operation. If this is the last or only command executed in the SST I/O operation, then the SST I/O operation and the task shall be terminated.

5.3.3 SST zero length operation

The initiator starts an SST zero length operation by issuing an ST Nop operation (see 6.1). An SST zero length operation is issued if the command involves no data transfer.

The SST zero length operation shall have an option payload including command control flags, addressing information, and the SCSI CDB as described in 0.

The SST zero length operation is the Execute Command service request and starts the I/O operation. The SST I/O operation in which the SST zero length operation is executed is identified by the initiator-specified tag qualified by the ST Virtual Connection. The tag shall be used as the ST I-id field for the SST zero length operation.

Initiator		Target
Initiate ST Nop sequence	ST Nop ⟶	
Write the SCSI Status to SPMR	⟵ ST Put Operation	

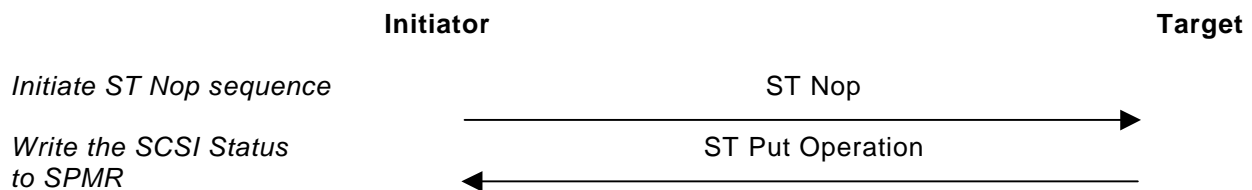**Figure 5 – SST zero length operation**

The device server shall return the Execute Command service response to the initiator by requesting that the target issue an ST Put sequence to the SPMR established in the SSTVC setup protocol (see 5.2.1). The address (Bufx and Offset) for this ST Put shall be computed from the initiator-specified tag used as the ST I-id for the SST zero length operation (see 6.3). This ST Put sequence shall contain the SCSI status. If an exception condition has been detected, then it shall also contain the SCSI REQUEST SENSE information and the SST Response information that describes the condition. The SST Status Put shall terminate the command. The SCSI logical unit determines whether additional linked commands will be performed in the SST I/O operation. If this is the last or only command executed in the SST I/O operation, then the SST I/O operation and the task shall be terminated.

5.3.4 Underrun residual condition handling

If the amount of data the target requires to be transferred as specified in the SCSI CDB is less than the length of the ST data transfer specified in the ST Request_To_Send or Request_To_Receive operation, either as a result of a successful or unsuccessful SCSI command completion, then this difference is called an underrun residual condition.

If an ST Write sequence in a task results in an underrun residual, then the target receives all data it has requested with outstanding ST Clear_To_Send operations before performing the SST Status Put operation which signifies the completion of the SCSI command.

If an ST Read sequence in a task results in an underrun residual and the initiator has sent ST Clear_To_Send operations for this ST Read sequence for data beyond the final block in which the target has transferred data, then the initiator shall ensure that all ST Clear_To_Send operations have been received by the target by issuing an ST End operation with an option payload that indicates the highest block number of ST Clear_To_Send operations issued (see 6.4). After all outstanding ST Clear_To_Send operations have been received, the target shall respond to the ST End operation with an ST End_Ack operation as described by the ST protocol. In order to support

this ST Clear_To_Send accounting protocol, an initiator shall only issue ST Clear_To_Send operations for a contiguous sequence of Blocks for SST Read operations.

5.3.5 Third-party SCSI commands

Certain third-party SCSI commands and parameters specify a 64-bit field that is defined to access other SCSI devices addressable from that port. These commands include COPY, RESERVE, and several others.

The ST protocol does not specify a specific address format. However, ST is usually run on an LLP that does specify an address format. Therefore, the address formats for third-party SCSI commands in the SST protocol are a function of the LLP address format. Address formats are beyond the scope of this standard.

5.4 SCSI task management

An application client requests a SCSI task management (TM) function when a task or some group of tasks need to be aborted or terminated.

SCSI TM functions are mapped onto SST and the underlying ST protocol as shown in table 2.

The SST zero length operation sent to initiate a TARGET RESET, LOGICAL UNIT RESET, ABORT TASK SET, CLEAR TASK SET or CLEAR ACA TM function shall be sent as the first and only operation in a new task. As with all other tasks, the target shall respond to these TM functions with an SST Status Put using the tag specified in the SST zero length operation that requested the TM function.

Table 2 – SCSI task management function mapping

SCSI	SST	ST	Required
ABORT TASK	End sequence	End sequence	Y
TARGET RESET	SST I/O operation with Target Reset bit set	Nop operation and Put operation	Y
LOGICAL UNIT RESET	SST I/O operation with LOGICAL UNIT RESET bit set	Nop operation and Put operation	Y
ABORT TASK SET	SST I/O operation with Abort Task Set bit set	Nop operation and Put operation	Y
CLEAR TASK SET	SST I/O operation with Clear Task Set bit set	Nop operation and Put operation	Y
CLEAR ACA	SST I/O operation with Clear Auto Contingent Allegiance bit set	Nop operation and Put operation	Y (if ACA is supported)

5.4.1 Abort Task

The SCSI Abort Task TM function may be used to terminate, prematurely, an outstanding SCSI command. The SST Abort Task protocol used to implement the SCSI Abort Task TM function ensures that in addition to aborting the SCSI command, all outstanding ST operations for that task have either reached their destination or been discarded before the SST Abort Task protocol is complete.

Once the SST Abort Task protocol is complete, both the initiator and target are free to reuse the tag and the ST sequence identifiers previously associated with the task.

It should be noted that at any time during the Abort Task TM functions, the ST Virtual Connection could be torn down. If this occurs, the target or initiator should reclaim any resources associated with the Virtual Connection and exit the Abort Task TM function.

For the Abort Task TM function, the Abort Task timeout should be the sum of the maximum time required for a target to perform internal functions associated with aborting the task and the maximum lifetime of an ST operation on the network. Generally, the Abort Task timeout may be the same as the task timeout for a task, since it should not take longer to abort a task than it would take to complete the task normally.

The following subclauses describe the processes used to abort an SST Write operation, an SST Read operation, and an SST zero length operation. Annex B contains example flow chart implementations for these processes.

5.4.1.1   Abort an SST Write operation

To abort an SST Write operation, the initiator shall:

a) stop sending data;

b) issue an ST End for the task to abort;

c) for every Block for which an ST Clear_To_Send has been received, ensure that an in-order STU with the ST Last Flag bit set to 1b, Interrupt Flag bit set to 1b, and Silent Flag bit set to 0b has been sent;

d) wait for an ST End_Ack for the task being aborted;

e) if an ST End_Ack is not received within the Abort Task timeout, determine the viability of the Virtual Connection (see 7.3) and:

   1) if the Virtual Connection is not viable, then tear down the Virtual Connection; otherwise,

   2) retry steps b through d an appropriate number of times after which the Virtual Connection shall be torn down.

The target shall issue an ST End_Ack if a task with the tag specified in the ST End is not in progress, otherwise:

a) for each ST Clear_To_Send operation issued, wait for a STU with the ST Last Flag bit set to 1b;

b) issue an ST End_Ack.

5.4.1.2   Abort an SST Read operation

To abort an SST Read operation, the initiator shall:

a) stop issuing ST Clear_To_Send operations;

b) issue an ST End operation indicating the Block number of the last ST Clear_To_Send operation in the option payload (see 6.4);

c) wait for a STU with the ST Last Flag bit set to 1b for every ST Clear_To_Send operation issued, or an ST End_Ack operation for the task being aborted;

d) if the operations awaited in step c are not received within the Abort Task timeout, then determine the viability of the Virtual Connection (see 7.3) and:

   1) if the Virtual Connection is not viable, then tear down the ST Virtual Connection, otherwise;

   2) retry steps b and c an appropriate number of times, after which the Virtual Connection shall be torn down.

The target shall issue an ST End_Ack if a task with the tag specified in the ST End is not in progress, otherwise:

a) stop sending data;

b) for each ST Clear_To_Send operation reported sent by the option payload of the ST End operation, ensure that an in-order STU with the ST Last Flag bit set to 1b, Interrupt Flag bit set to 1b, and Silent Flag bit set to 0b has been issued;

c) issue an ST End_Ack.

5.4.1.3 Abort an SST zero length operation

To abort an SST zero length operation, the initiator shall:

a) issue an ST End;

b) wait for an ST End_Ack operation for the task being aborted;

c) if the awaited ST End_Ack is not received within the Abort Task timeout, then determine the viability of the Virtual Connection (see 7.3) and:

    1) if the ST Virtual Connection is not viable, then tear down the ST Virtual Connection, otherwise;

    2) retry steps a and b an appropriate number of times, after which the Virtual Connection shall be torn down.

The target shall issue an ST End_Ack in response to the ST End sent by the initiator.

6 SST protocol operation formats

6.1 ST Nop operations in SST

SST uses the 16-bit Param field of the ST Nop operation as an operation code as shown in table 3.

Table 3 – SST Nop operation codes

Definition	Value
Request_Zero_Length	0h
SSTVC_Parameters	1h

The 32-bit S_id field of an SST zero length operation contains the tag.

The Param and S_id fields of the ST Nop operation are ST Opaque data. The remainder of the ST Nop Opaque data fields in the SST zero length operation (B_id, Bufx, Offset, Sync, B_num, and D_id) are unused and should be set to zero (see ST). The option payload of the SST zero length operation is described in 0.

6.1.1 Option payload for SSTVC_Parameters Operation

Figure 6 shows the format of the option payload data exchanged during an SSTVC_Parameters operation.

     Working Draft SCSI on Scheduled Transfer (SST)

			Byte
ST_OPTION_CODE	LENGTH	MAX_TARGET	00-03
FLAGS			04-07
SPMR_SIZE			08-11

Figure 6 – Option payload exchanged during an SSTVC_Parameters operation

6.1.1.1 ST_OPTION_CODE

ST_OPTION_CODE set to 03h indicates that the ST option payload field is valid and contains a ULP parameter.

6.1.1.2 LENGTH

LENGTH set to 0Ch indicates that this ST option payload field is 12 bytes long.

6.1.1.3 MAX_TARGET

MAX_TARGET is the maximum supported value of the TARGET field (see 6.2.3) in the option payload for an ST Request_To_Send, an ST Request_To_Receive, or an SST zero length operation. Note that the maximum number of targets is MAX_TARGET+1.

6.1.1.4 FLAGS

The FLAGS field contains a number of supported and required feature flags as shown in figure 7 and described in the following text.
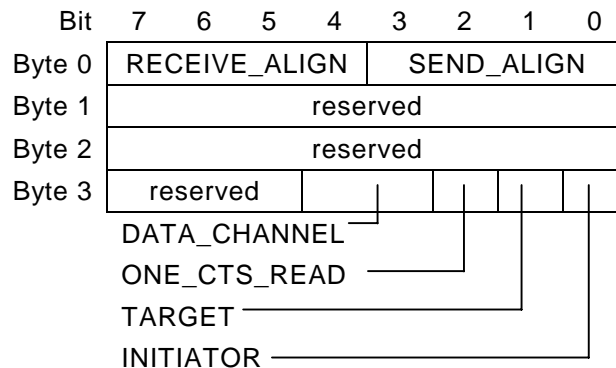

Figure 7 – Option payload Flags for Connection operations

RECEIVE_ALIGN is the log, base 2, of the minimum required alignment of the offset parameter of any ST Data operations sent to the SST Virtual Connection endpoint specifying the RECEIVE_ALIGN value. This requires that an SST endpoint shall never issue an ST Clear_To_Send operation with an initial offset that does not conform to the RECEIVE_ALIGN value specified during the SST Connection setup.

SEND_ALIGN is the log, base 2, of the minimum required alignment of the offset parameter of any ST Clear_To_Send operations sent to the SST Virtual Connection endpoint specifying the SEND_ALIGN value. Alignment values are likely to reflect a word size in an alignment-restricted implementation (e.g. 4 bytes, or 8 bytes).

DATA_CHANNEL indicates the ST Data Channel to request for the SST SPMR.

ONE_CTS_READ set to 1b indicates that a target requires the initiator to issue only a single ST Clear_To_Send operation to cover the entire data transfer of an SST Read operation.

TARGET set to 1b indicates that the SST endpoint is capable of performing SCSI target functions.

INITIATOR set to 1b indicates that the SST endpoint is capable of performing SCSI initiator functions.

6.1.1.5   SPMR_SIZE

The SPMR_SIZE field indicates the number of 512-byte SPMR segments a target shall request that an initiator expose. The size of the SPMR, in bytes, shall be calculated as:

$$\text{SPMR\_SIZE} * 512 \text{ bytes}$$

One SPMR segment is needed for each outstanding task. Thus, the number of SPMR segments requested will reflect the desired level of task concurrency allowed on an SSTVC.

6.1.1.6   SSTVC Reject reason codes

If the parameters of an SSTVC are unacceptable, then the ST Virtual Connection shall be torn down with an appropriate reason code (as shown in table 4) in the option payload of the ST Request_Disconnect operation.

Table 4 – SST Connection Reject reason codes

Definition	Value
Single CTS Read unsupported	8000h
RECEIVE_ALIGN value unsupported	8001h
SEND_ALIGN value unsupported	8002h
Target function unsupported	8003h
Initiator function unsupported	8004h
SPMR not established	8005h
SPMR unaligned	8006h
Busy (no resources)	0004h

6.2   Option payload for SST command operations

The option payload for ST Request_To_Send, ST Request_To_Receive, and SST zero length operations is organized as shown in figure 8.

Byte

ST_OPTION_CODE	LENGTH	TARGET	00-03
LUN			04-07
			08-11
CNTL			12-15
SCSI CDB			16-19
			20-23
			24-27
			28-31

Figure 8 – Option payload for SST command operations

6.2.1   ST_OPTION_CODE

ST_OPTION_CODE set to 03h indicates that the ST option payload field is valid and contains a ULP parameter

6.2.2   LENGTH

LENGTH set to 20h indicates that the ST option payload field is 32 bytes long.

6.2.3   TARGET

TARGET selects one of a set of individual SCSI targets addressable through a single SST Virtual Connection.   The maximum allowable value of the TARGET field is established in the SST connection setup protocol.

6.2.4   LUN

The SCSI Logical Unit Number (LUN) is the address of the desired logical unit in the attached subsystem. The LUN field is specified by ANSI X3.270.

6.2.5   CNTL

The CNTL field contains SCSI control flags and control bits as defined in ANSI X3.270 and are organized in SST as shown in figure 9.
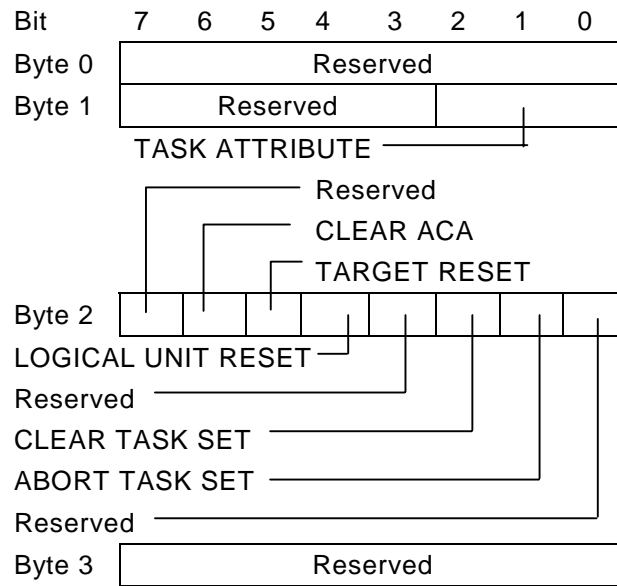
**Figure 9 – CNTL field definitions for SST
command operations**

6.2.5.1 Task Codes, Byte 1

TASK ATTRIBUTE shall be selected as defined in ANSI X3.270, shown in table 5 as an aid to the reader.

**Table 5 – TASK ATTRIBUTE definitions for
SST Command operations**

Value, bits 2-0	TASK ATTRIBUTE
000b	SIMPLE_Q
001b	HEAD_OF_Q
010b	ORDERED_Q
100b	ACA_Q
101b	UNTAGGED
others	reserved

6.2.5.2 Task management flags, byte 2

If any TM flag is set to 1b, then the CDB and CDB related CNTL flags (task codes and execution management codes) are not valid and shall be ignored. No more than one TM flag shall be set to 1b in any SST zero length operation. No TM flags shall be set for an ST Request_To_Send or ST Request_To_Receive operation.

6.2.6 SCSI CDB

The SCSI CDB field shall contain the actual CDB to be interpreted by the addressed logical unit. The maximum CDB length shall be 16 bytes. The CDB is not valid and shall be ignored if any TM flag is set to 1b. The CDB command byte (CDB byte 0) shall be the first byte following the CNTL

parameter. Subsequent CDB bytes shall be stored in subsequent bytes of the option payload. Bytes beyond the last byte of the CDB are not defined by SST, shall be ignored by the target, and should be set to zero.

6.3   SST Status Put sequence

The SST Status Put sequence shall be directed to an address (Bufx/Offset) which is computed as:

Status Put Bufx = address ÷ I-Bufxsize

Status Put Offset = address modulo I-Bufxsize

where:

I-Bufxsize = $2^{\text{Initiator Bufsize}}$, and

address = (SPMR initial Bufx × I-Bufxsize + SPMR initial offset) + (512 × tag).

The tag shall be sent in the ST S_id field (opaque in the ST specification) of the ST Data operations of the SST Status Put.

The SST Status Put sequence shall consist of a single ST Data STU with the entire SST Status Put payload.

The payload of the SST Status Put sequence shall be as shown in figure 10.

	Bytes
Reserved	00-03
	04-07
STATUS	08-11
RESID	12-15
SNS_LEN	16-19
RSP_LEN	20-23
RSP_INFO (0, 4, or 8 Bytes)	
SNS_INFO (n Bytes)	

**Figure 10 - Payload Parameters for an SST
Status Put**

6.3.1   STATUS

Figure 11 shows the format of the STATUS field.
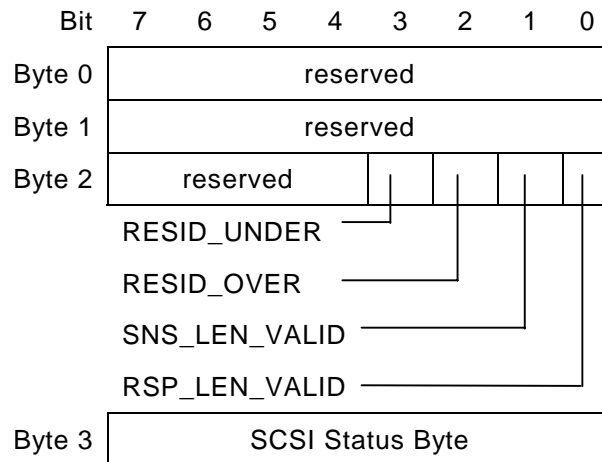
Figure 11 – STATUS field definitions for SST
Status PUT

RESID_UNDER set to 1b indicates that the RESID field is valid and contains the count of bytes that were expected to be transferred, but were not transferred.

RESID_OVER set to 1b indicates that the RESID field is valid and contains the count of bytes that could not be transferred because the ST transfer length was not sufficient.

SNS_LEN_VALID set to 1b indicates that the SNS_LEN field is valid and contains the count of bytes in the SNS_INFO field.

RSP_LEN_VALID set to 1b indicates that the RSP_LEN field is valid and contains the count of bytes in the RSP_INFO field.

Byte 3 contains the status byte from the SCSI logical unit. The status byte codes are defined by ANSI X3.270.

6.3.2   RESID

If RESID_UNDER is set to 1b or RESID_OVER is set to 1b, then the RESID field contains a count of the number of residual data bytes that were not transferred for this SCSI command.

If RESID_UNDER is set to 1b, then a transfer that did not fill the buffer to the expected displacement specified by the ST transfer length was performed and the value of RESID is a number equal to:

    ST transfer length – highest offset of any byte transmitted

A condition of RESID_UNDER may not be an error for some devices and some commands.

If RESID_OVER is set to 1b, then the transfer was truncated because the data transfer required by the SCSI command extended beyond the transfer length specified in the ST Request_to_Send or Request_to_Receive operations. Those bytes that could be transferred without violating the ST transfer length value may be transferred. RESID is a number equal to:

    (Transfer length required by command) – ST transfer length

If a condition of RESID_OVER is detected, then the termination state of the SST I/O operation is not certain. Data may or may not have been transferred and the SCSI status byte may or may not provide correct command completion information.

If the RESID_UNDER and the RESID_OVER bits are 0b, then the RESID field is not meaningful and may contain any value.

6.3.3   SNS_LEN

If SNS_LEN_VALID is set to 1b, then the SNS_LEN field specifies the number of valid bytes of SNS_INFO.

If SNS_LEN_VALID is set to 0b, then the SNS_LEN field is not valid and no SNS_INFO is provided.

The SNS_LEN field shall be included in the SST Status Put.

6.3.4   RSP_LEN

If RSP_LEN_VALID is set to 1b, then the RSP_LEN field specifies the number of valid bytes of RSP_INFO. The number of valid bytes shall be 0, 4, or 8. Other values of length are reserved for future standardization.

RSP_LEN set to 0h specifies that no bytes of response information are being provided.

If RSP_LEN_VALID is set to 0b, then the RSP_LEN field is not valid and no RSP_INFO is provided.

The RSP_LEN field shall be included in the SST Status Put.

6.3.5   RSP_INFO

The RSP_INFO field contains information describing protocol failures detected during the execution of an SST I/O operation. RSP_INFO does not contain SCSI logical unit error information since that is contained in the SNS_INFO field as specified in 6.3.6. The RSP_INFO field shall contain valid information if the target detects any of the conditions indicated by a RSP_CODE in table 6. The format of the RSP_INFO field is shown in figure 12.

Bit	7	6	5	4	3	2	1	0
Byte 0	reserved							
Byte 1	reserved							
Byte 2	reserved							
Byte 3	RSP_CODE							

Figure 12 – RSP_INFO field definitions for SST Status PUT

Table 6 – RSP_CODE definitions for SST Status PUT

RSP_CODE definition	Value
No failure or task management function complete	00h
reserved	01h
SST CTS or RTS payload fields invalid	02h
reserved	03h
Task management function not supported	04h
Task management function failed	05h
Nonexistent target	06h
Busy (no resources)	07h
reserved	08h – FFh

6.3.6 SNS_INFO

The SNS_INFO field contains the information specified by ANSI X3.270 for presentation by the REQUEST SENSE command. The SNS_INFO shall be presented when the SCSI status byte of CHECK CONDITION or COMMAND TERMINATED is presented as specified by ANSI X3.270. SST shall implement the autosense mechanism as specified in ANSI X3.270.

6.4 Option payload for ST End operations

When the ST End operation is required by the SST protocol to specify the block number of the last ST Clear_To_Send operation issued, the ST End operation option payload shall be as shown in figure 13.

Byte

ST_OPTION_CODE	LENGTH	reserved	00-03
B_NUM			04-07

Figure 13 – Option payload for ST End operations

6.4.1 ST_OPTION_CODE

ST_OPTION_CODE set to 03h indicates that the ST option payload field is valid and contains a ULP parameter.

6.4.2 LENGTH

LENGTH set to 08h indicates that the ST option payload field is 8 bytes long.

6.4.3 B_NUM

B_NUM indicates the ST Block number of the last ST Clear_To_Send operation sent by the data Destination.

Since the SST protocol requires in-order request of ST data Blocks, the ST Block number of the last ST Clear_To_Send operation permits the data source to ensure that all ST Clear_To_Send operations for a task have been received before reusing the ST sequence identifiers associated with the task. This permits the endpoints to avoid aliasing of ST Clear_To_Send operations across tasks when sequence identifiers are reused.

7   Error recovery procedures

7.1   Error recovery overview

The SST I/O operation timeout is used to detect the loss of any ST operation during a task. SST does not use any of the optional ST data transmission timeout pairs defined in ST.

SST does not use ST sequence identifiers for antialiasing. The ST initiator sequence identifier is used as the tag, which specifies the SPMR region into which to put SCSI status information, so this mechanism is not available to perform rapid recovery from loss of ST operations.

In certain exception cases, such as underrun residual, SST defines clean-up mechanisms to rapidly ensure that no ST operations remain outstanding which could alias with future SST operations on the same SSTVC, using the same sequence identifier.

The remaining exception cases are cleaned up using the SST I/O operation timeout to ensure that no ST operations remain outstanding which could alias with future SST operations on the same SSTVC.

If an implementation desires more rapid cleanup than is offered by the SST I/O operation timeout mechanism, that implementation may take advantage of the antialiasing features of ST by closing the SSTVC and opening a new one.

7.2   SSTVC Setup Errors

Until the ST Memory_Region_Available operation is received, an SSTVC Responder shall respond to any task initiating operation (ST Request_To_Send, ST Request_To_Receive or SST Request_Zero_Length) by tearing down the SSTVC with a Reject reason code of 'SPMR not established' (see 6.1.1.6).

If the initial offset returned in the ST Memory_Region_Available operation for the SST SPMR is not a multiple of the Status Segment Size, i.e. 512 bytes, then an SSTVC Responder shall respond to any data transfer request by tearing down the SSTVC with a Reject reason code of 'SPMR unaligned' (see 6.1.1.6).

7.3   Determining the viability of a VC

A target shall implement ST Virtual Connection keep-alive sequences as described in the ST standard. An initiator may implement ST Virtual Connection keep-alive sequences as described in the ST standard.

If an ST Virtual Connection is discovered to be dead by a failure of the keep-alive test, then all tasks on the Virtual Connection shall be closed and their resources reclaimed, as well as the resources for the Virtual Connection itself.

Determining the viability of a Virtual Connection consists of performing an ST Request_State/Request_State_Response sequence with an appropriate number of retries.

For a low loss medium, a single retry would be an appropriate default for any step in the SST Abort Task protocol that requires retries.

7.4   SST I/O operation timeouts

An initiator shall maintain timeouts on all outstanding device and task management operations from the time an ST Request_To_Send, ST Request_To_Receive or SST zero length operation is sent until the final Data and SST Status Put operations are received.

The SST I/O operation timeout should be at least the sum of:

— the maximum time required to perform the SCSI command and transfer the associated data; and,

— the maximum time an ST operation can remain in the network.

The actual determination of the SST I/O operation timeout value is beyond the scope of this standard.

This timeout ensures that no ST operations remain in the network, or will be subsequently generated by a target for a task, which guarantees that an initiator may reuse the tag associated with the task immediately.

If an SST I/O operation timeout expires, then the initiator shall first determine the viability of the SST Virtual Connection using an ST Request_State/Request_State_Response sequence. If an ST Request_State_Response operation is not received within an appropriate period of time, then the ST Request_State/Request_State_Response sequence shall be retried an appropriate number of times after which the Virtual Connection shall be torn down and its resources reclaimed (see 6.1).

If an ST Request_State_Response is received, then the operation that timed out shall be aborted with the SST Abort Task protocol as described in 5.4.1, Abort Task.

A target shall not maintain timeouts on outstanding tasks. The target resources for a task shall only be reclaimed when either:

— the target completes the task; or

— the initiator ends the task with the SST Abort Task protocol; or

— the SSTVC is torn down.

Annex A

(Normative)
SCSI on ST Profile

This annex summarizes limitations and requirements within the Scheduled Transfer protocol (*ANSI NCITS.337-2000*, Information Technology - Scheduled Transfer (ST)) in order to facilitate interoperability for SST implementations.

A.1 Applicability and use of this annex

Since the nature of this annex is a profile, the usual definitions of the following words do not apply.

Prohibited: If a feature is prohibited, then it shall not be used by compliant SST implementations.

Required: If a feature or parameter value is Required, it means that it shall be used between compliant SST implementations. Compliant implementations are required to implement the feature.

Allowed: If a feature or parameter value is Allowed, it means that it may be used between compliant SST implementations. Compliant implementations are not required to implement the feature, but if they do, the feature shall be used as described in this document. Typically, the potential user of a feature may determine if the potential recipient supports that feature via a Required discovery process.

Invocable: If a feature or parameter value is Invocable, it means that it may be used between compliant SST implementations. Compliant implementations are required to implement the feature, and make available the use of the feature. Invocable is different than Allowed or Required in that an originator may invoke the feature if needed, but the originator is not required to invoke it, and may never need to.

Tables in the following clauses list features described in the ST protocol. These tables indicate whether the feature is Required, Prohibited, Invocable, or Allowed for compliance with this profile; or whether a parameter is Required to be a particular value or limited range of values for compliance with this profile.

Features or parameters that are not listed do not affect the interoperability of SST implementations.

The following legend is used for table entries in these clauses:

'P' the implementation is Prohibited from using the specified feature

'R' the implementation is Required to support the specified feature

'A' use of the specified feature is Allowed

'I' the implementation may Invoke the specified feature

A.2 Compliance with ST

Any ST requirement not specifically constrained by this profile shall be supported.

Any ST option that is not constrained by this profile is allowable.

Any allowable ST parameter value that is not constrained by this profile may be any value allowed by ST.

A.3 Environmental requirements beyond ST

This profile and the ST protocol make many of the same assumptions about underlying protocol layers. For example, it is assumed that there is a mechanism whereby ST Messages are delivered from the sender to the receiver.

A.4 Profile Settings

Table 1 lists general characteristics and assigns behavior(s) or parameter value(s).

Table 2 lists connection characteristics and assigns behavior(s) or parameter value(s).

Table 3 lists data transfer characteristics and assigns behavior(s) or parameter value(s).

Table A.1 - General behavior and options

Parameter or Characteristic	Initiator	Target	Notes
Timeout on op pairs	P	P	use single timeout per task instead
Heartbeat Timer	A	R	
Push Semantics	R	P	data transfer only from initiator to target
Pull Semantics	R	P	data transfer only from target to initiator
Expose Persistent Memory	R	P	only Put, only from target to initiator
Checksums for non-data	A	A	
Option Payloads	R	P	initiator sends, target receives, only on RTR, RTS, End, Nop.
Slot accounting	A	A	allowed, but expect to use SCSI command flow control model (queue depth + busy responses)
Nop	R	P	Used for SST zero length operations

Table A.2 - Connection behavior and options

Parameter or Characteristic	Initiator	Target	Notes
Party Line	P	P	
Option Payloads during connection	P	P	
Request State Response	A	R	Used only to determine SSTVC viabilitiy

Table A.3 - Data transfer behaviors and options

Parameter or Characteristic	Initiator	Target	Notes
Variable Block Sizes	R	R	
Consistent Block Sizes	P	P	
Silent on any STU other than last	R	R	
Silent on last STU	P	P	
Interrupt flag on last STU	R	R	
Checksums for Data STU Blocks	A	A	
Out Of Order Blocks	P	P	
Buffer size > Block size	I	I	
Block size > Buffer size	I	I	
STUs not in ascending order	P	P	
STU Size minimum	512	512	
STU Size maximum	1MB	1MB	
Block size minimum	512	512	
Block size maximum	32MB	32MB	
Buffer size minimum	512	512	
Buffer size maximum	512MB	512MB	
Block retransmission	P	P	entire task retried on error
Opaque Data in Data Operations	R	R	for status Put from target to initiator
Request State Response	P	P	Used only to determine VC viabilitiy (see table A.2)
Unlimited Transfer size	P	P	
Bi-directional	A	A	subject to declared SCSI roles in VC setup
Data Channel Usage of VC1	I	I	DC selected during VC setup
Data Channel Usage of VC2	I	I	DC selected during VC setup
Data Channel Usage of VC3	I	I	DC selected during VC setup
More than 8 CTS	P	P	
Striping	P	P	
Non-zero offsets	A	A	offset alignment restricted by parameter during VC setup

Annex B

(Informative)
Abort Task Flow Charts

This Annex B contains flow charts to illustrate the algorithms for the Task Management functions that Abort an SST Write operation, Abort and SST Read operation, and Abort an SST zero length operation presented in subclause 5.4.1.

B.1 Abort an SST Write operation

Figure B.1 illustrates, in flow chart format, an implementation of the algorithm to Abort an SST Write operation as described in subclause 5.4.1.1.
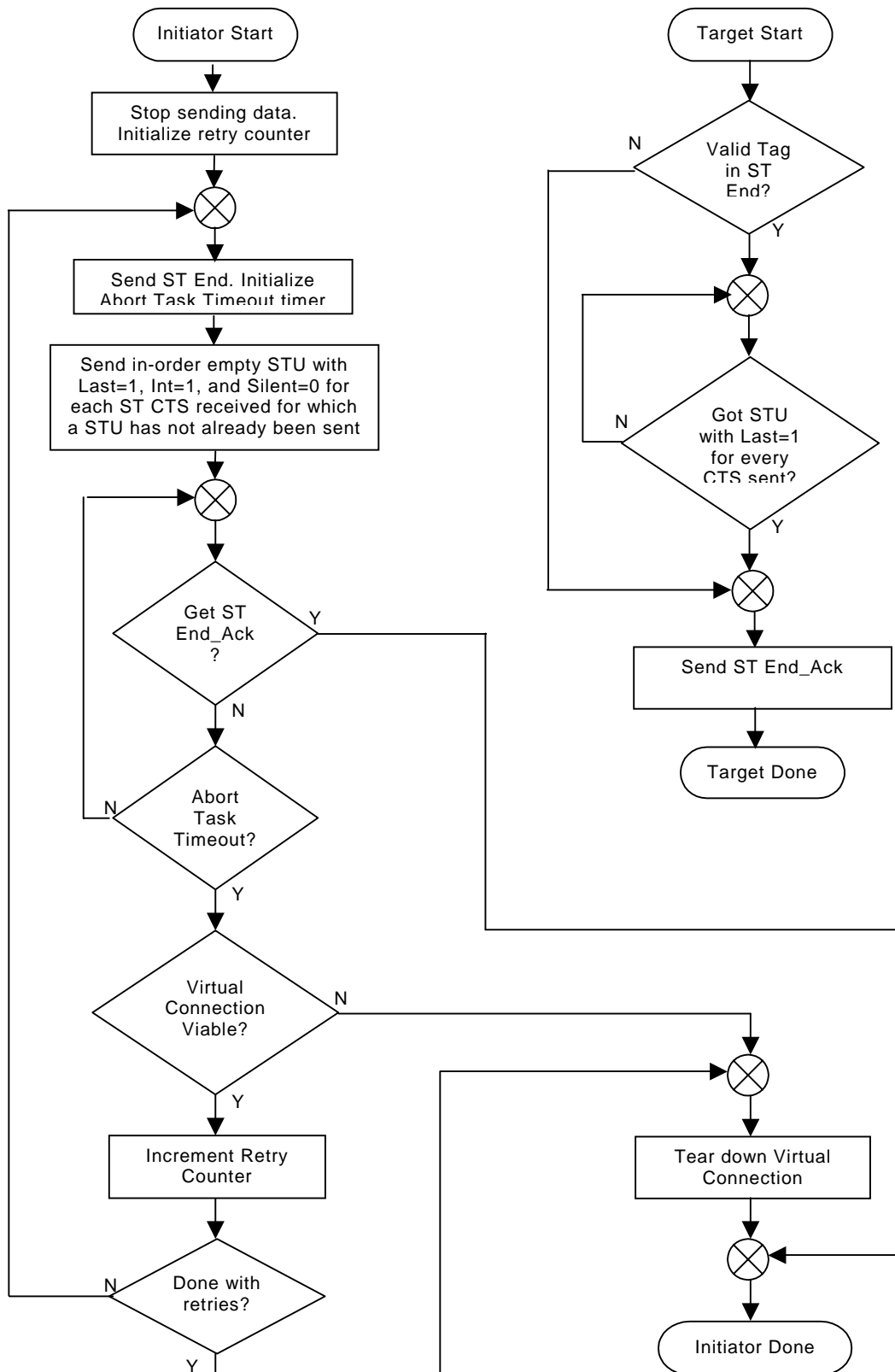
Figure B.1 Abort an SST Write operation

B.2 Abort an SST Read operation

Figure B.2 illustrates, in flow chart format, an implementation of the algorithm to Abort an SST Read operation as described in subclause 5.4.1.2.
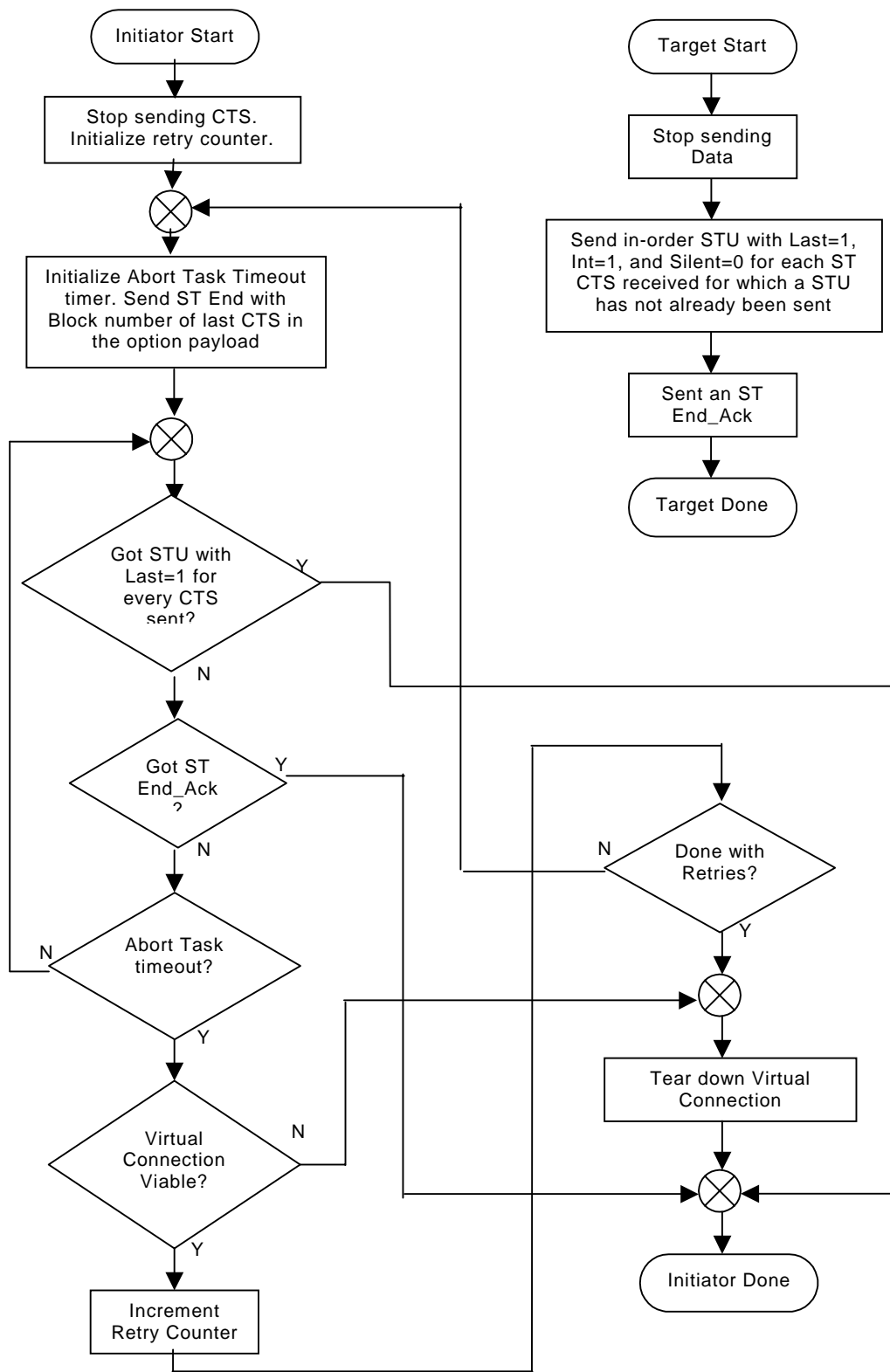
Initiator Start

Stop sending CTS.
Initialize retry counter.

Initialize Abort Task Timeout
timer. Send ST End with
Block number of last CTS in
the option payload

Got STU with
Last=1 for
every CTS
sent?

Got ST
End_Ack
?

Abort Task
timeout?

Virtual
Connection
Viable?

Increment
Retry Counter

Target Start

Stop sending
Data

Send in-order STU with Last=1,
Int=1, and Silent=0 for each ST
CTS received for which a STU
has not already been sent

Sent an ST
End_Ack

Target Done

Done with
Retries?

Tear down Virtual
Connection

Initiator Done

Figure B.2 Abort an SST Read operation

B.3 Abort an SST zero length operation

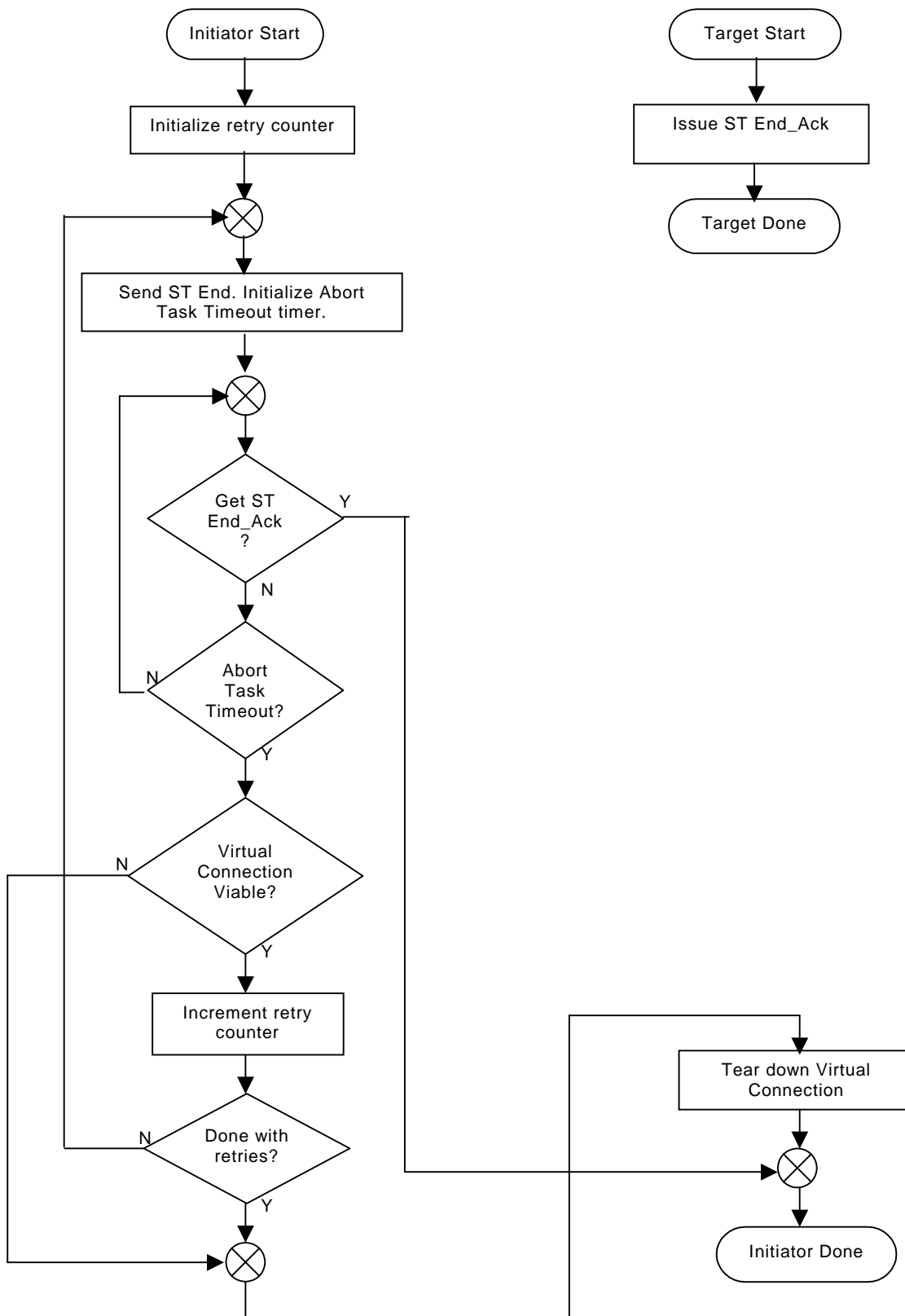Figure B.3 illustrates, in flow chart format, an implementation of the algorithm to Abort an SST zero length operation as described in subclause 5.4.1.3.

Figure B.3 Abort an SST zero length operation